

1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, tens of thousands of emails and detailed financial data for top executives entered into the public record. The goal of this project is to build a person of interest (POI) identifier based on Enron financial and email data. List of persons of interest in the fraud case was hand-generated.

Dataset description

- 146 data points, each point corresponds to a person
- 21 features
- Target variable 'poi' takes value 1 for the record corresponding to POI and 0 otherwise.
- Target variable has highly unbalanced distribution in the dataset, overall only 12% of records correspond to POIs (18 persons).
- Testing of connections between features confirmed that financial data is presented by total payments, total stock values and their constitutions.

Missing values

Excluding target variable 'poi' and email address, there are 19 features which can be used for classification. Among these variables a few have a low percent of non-missed values (>30 % of values are missed)

Feature	Percent of non-missed values
deferral_payments	26.71%
restricted_stock_deferred	12.33%
director_fees	11.64%
loan_advances	2.74%

I did not to use loan_advances, since it is missed almost in all records. Since it is a constitution of total payments I have added existing values to 'other' feature.

Outliers & Errors

Data contains outliers and some of them correspond to famous participants of Enron scandal, so these data points are definitely worth keeping in analysis. However, some outliers were errors, detected errors are listed in the table below.

Record	Problem description	Detection	Action
"TOTAL"	Not a person	Distribution check	Deleted
"LOCKHART EUGENE E"	No data for a record	Check of missed values	Deleted
"THE TRAVEL AGENCY IN THE PARK"	Not a person	Check name patterns	Deleted
"BELFER ROBERT"	Shifted data	Cross check with total payments and total stock value	Fixed
"BHATNAGAR SANJAY"			

Cleaning process is implemented in function clean, impl/preprocess.py.

One of the main point of concern for this analysis is unbalanced allocation across classes in the dataset (12% vs 88%) Note, that prediction like 'not a POI' for any case already gives 88% accuracy. That is why accuracy is not suitable for unbalanced problems. In cross validation simple random splits can also be misleading, more complex techniques like stratified splits (ratio of POI and non-POI is the same during training and testing) can be more useful here.

In further analysis I will address mentioned issues. I will focus on such metrics like precision, recall and F1-score. In cross validation I will use Stratified shuffle splits.

2. *What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.*

I used automated feature selection function SelectPercentile. This function ranks all features according to specific score function and keeps only specified percent of them. I have included feature selection as preprocessing step in a classification pipeline and tuned its parameters along with classification parameters. I considered two score functions: one based on ANOVA F statistic and one based on mutual information, four variants of percentage cuts: 30%, 50%, 70%, 100%. These grid is quite wide and can help to see general trends. If necessary, it can be detailed around specific range. Note, that percentage cut 100% means that no filtering is performed and all features are used.

I have engineered some new features on the basis of existing, description is presented in the table below.

Features	Description	Reasoning
shared_receipt_with_poi_perc from_poi_to_this_person_perc	Fraction of received and shared messages with POI in all received messages	New variables can possibly help to detect if a person has regular communications with POIs
from_this_person_to_poi_perc	Fraction of mails sent to POI in all sent mails	
gross_payments	total payments + 2 * abs(deferred income)	New variables account for future payments as well. They can provide more reasonable comparison of payments.
gross_stock_value	total stock value + 2 * abs(restricted stock deferred)	
salary_perc bonus_perc etc.	Total payment components (salary, bonus, etc) calculated as fractions of gross payments	Is it suspicious if, for example, most payments go as bonuses or "other"?
exercised_stock_options_perc restricted_stock_perc etc.	Total stock values component s calculated as fractions of gross stock value	New variables are measured on a percentage scale and can answer questions of this kind.

After adding new features, I reran some algorithms on the extended set of features using automated selection function SelectPercentile. Both on original and extended feature sets Logistic Regression showed the best performers among all tested algorithms. On extended feature set F1-score of Logistic regression increased on 0.05 points (see ver4 and ver5 in the table in the next section).

Note that for SelectPercentile I used the same wide grid for percentage cuts. I did not go forward with detalization of the grid, because regularization incorporated in Logistic Regression algorithm already performs some additional penalization of insignificant features.

3. *What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?*

As a final classifier I used a three step algorithm: min/max scaling, automated feature selection with SelectPercentile and Logistic Regression. Along with Logistic Regression I have tried three other classifiers: Naïve Bayes, Decision Tree and SVM. Logistic Regression outperformed other algorithms both on original and extended feature sets. SVM showed similar performance in sense of F1 score, however, it had significantly lower recall.

Cross Validation Results

Version	Accuracy	Precision	Recall	F1
ver1.2(Naïve Bayes)	0.809	0.3972	0.395	0.3479
ver2(Decision Tree)	0.731	0.2863	0.64	0.3863
ver3(SVM)	0.6486	0.2764	0.9	0.4198
ver4(Logistic Regression)	0.7345	0.3153	0.7	0.4204
<i>Using extended data set with engineered features</i>				
ver5(Logistic Regression)	0.7672	0.3637	0.725	0.471
ver7(Decision Tree)	0.819	0.3948	0.4475	0.3979
ver8(SVM)	0.8176	0.4177	0.615	0.4791

Detailed analysis is presented in [analysis/POI classification.html](#)

4. *What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).*

Algorithm performance heavily depends on selected parameters, that is why it is important to investigate different sets of parameters and carefully define default values. I have tuned parameters of classifiers as well as some parameters of feature selection algorithm. I used grid search to find best set of parameters, performance was estimated via F1 score by cross validation using stratified shuffle splits.

5. *What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?*

Validation is the process of accessing algorithm performance on a new unseen data (in other words, in real life conditions), validation is intended to prevent over fitting and select algorithms which are able to generalize the data. A classic mistake is to use the same data for fitting and validation or evaluation. Validation should be always made on a separate data set or using cross-validation. In both cases, algorithm is fitted on the one part of the data, but performance is estimated on the other data which was not used for fitting.

I used grid search to find best parameters of the model, performance was estimated by cross validation using stratified shuffle splits. Note that validation and final evaluation was performed on the whole dataset. Often it is recommended to leave some part of the data for final evaluation, however, in case of very small datasets it is not always possible. Final evaluation was performed again with stratified shuffle splits, but with number of splits and random state different from validation step.

6. *Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance.*

As an evaluation metric for grid search I used F1 score, since it takes into account both precision and recall. For algorithm comparison I used four metrics: accuracy, precision, recall and F1.

Final algorithm has the following performance (according to tester.py):

Accuracy: 0.75969 Precision: 0.33056 Recall: 0.72400

Final algorithm has quite high recall, it means nearly 72% of persons of interest will be detected. Precision is significantly lower, among all persons identified as POI by the algorithm, only 33% are real POI. Meaning of accuracy is straightforward, nearly 76% of predictions are correct. Note that having high recall can be beneficial for this case since it is important to detect as many POI as possible.